APP #18
2700 6/6/02
Chong

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: **DAVID HETHERINGTON ET AL.**	§	Docket No. **AT9-98-160**
Serial No. **09/211,803**	§	Examiner: **F. COBY**
Filed: **December 15, 1998**	§	Art Unit: **2171**
For: **METHOD, SYSTEM & COMPUTER PRODUCT FOR STORING TRANSLITERATION AND/OR PHONETIC SPELLING INFORMATION IN A TEST STRING CLASS**	§	

APPEAL BRIEF

Assistant Commissioner of Patents
Washington, D.C. 20231

Sir:

This Brief is submitted in triplicate in support of the Appeal in the above-identified patent application.

CERTIFICATE OF MAILING
37 CFR § 1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner of Patents, Washington, D. C. 20231 on the date below.

May 13, 2002 Debbie Moore
Date

REAL PARTY IN INTEREST

The real party in interest in the present Appeal is International Business Machines Corporation (*IBM*), the Assignee of the present Application, as evidenced by the Assignment set forth and recorded at Reel 9666, Frame 0580.

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellant, the Appellant's legal representative, or assignee, which directly affect or would be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-25 stand finally rejected by the Examiner as noted in the Advisory Action dated February 11, 2002.

STATUS OF AMENDMENTS

No amendment to the claims was entered subsequent to the Final Rejection.

SUMMARY OF INVENTION

A fundamental problem in multinational computing environments which need to display data in multiple human languages is that a spoken word generally encapsulates information in multiple aspects or attributes, such as through the word's meaning, from context, and/or from inflection. When reduced to a visual or electronic representation for manipulation or display in a data processing system, the word may lose some attributes and much of the associated meaning. Most importantly for data processing systems, a visual representation of a word may give no clues as to the correct translation or pronunciation of the word or the proper placement of a word within a specified sort order. The present invention may be employed to address this problem (Application page 10).

As set forth in the present Specification at Page 10, line 11, *et seq.*, the preset invention may be utilized to encapsulate identification, meaning, and pronunciation information for a text string. In short, the present invention may be described as ***a text string data structure*** comprising a plurality of discrete fields. These fields are utilized to store the visual representation, pronunciation, and sorting information for a word. The present invention may be described with greater detail with reference to Figure 2 and its accompanying description.

Referring to Figure 2, a diagram of a multi-field text string class employed in encapsulating transliteration and/or phonetic spelling information in accordance with a preferred embodiment of the present invention is depicted. A fundamental problem in multinational computing environments which need to display data in multiple human languages is that a spoken word generally encapsulates information in multiple aspects or attributes, such as through the word's meaning, from context, and/or from inflection. When reduced to a visual or electronic representation for manipulation or display in a data processing system, the word may lose some attributes and much of the associated meaning. Most importantly for data processing systems, a visual representation of a word may give no clues as to the correct translation or pronunciation of the word or the proper placement of a word within a specified sort order. International String ("IString") class 202 may be employed to address this problem.

IString class 202 is preferably a Java class similar to the Java String class, which behaves like the String class by including similar methods. Most of the original behavior of the String class should be preserved, with additional functionality added and utilized only as needed. IString class 202 is a datatype which captures some of the meaning of spoken words which is normally lost when the word is reduced to a visual representation. IString class 202 is preferably utilized for all object names and system messages within a system.

The IString class 202 structure includes three different strings for each name, message, data or text object: a baseString 204, a sortString 206, and an altString 208. BaseString 204 is the string within IString class 202 employed by default in the user interface display and may contain any text, usually the original text entered by the user in the local language where the IString object is created. SortString 206 may also be any text and is employed to allow correct sorting of non-phonetic languages and languages which are difficult to sort based only on the binary value of baseString 204. AltString 208 may be any text but should conventionally be filled with a latin character set representation of the pronunciation of the data contained in baseString 204. Thus, IString class 202 includes the original text (baseString 204), a sort key (sortString 206), and a pronunciation key (altString 208) for object names, system messages, and other data.

ISSUE ON APPEAL

1. Is the Examiner's rejection of Claims 1-3, 12-13 and 19-20 under 35 U.S.C. §102(e) as being anticipated by *Li, et al.*, U.S. Patent No. 6,205,418, well founded?

2. Is the Examiner's rejection of Claims 4-11, 14-18 and 21-25 under 35 U.S.C. § 103(a) as being unpatentable over *Li, et al.*, U.S. Patent No. 6,205,418, in view of Renegar, U.S. Patent No. 6,024,571, well founded?

GROUPING OF CLAIMS

For purposes of this Appeal, Claims 1-9, 11-15 and 17-20 stand or fall together as a first group. Additionally, Claims 10 and 16 stand or fall together as a second group.

ARGUMENT

Regarding the first group of claims, the Examiner has rejected Claims 1-3, 12-13 and 19-20 under 35 U.S.C. §102(e) as being anticipated by *Li, et al.*, U.S. Patent No. 6,205,418. That rejection is not well founded and it is should be reversed.

Li, et al., discloses a system and method for providing a multiple language capability in a computer-based target application. The invention in *Li* utilizes a language and translation database to accomplish its objectives. More specifically, the language database contains a language name and language code (Col. 7, line 61-64) and the translation table contains entries composed of a language code as listed in the language database, a translation code, and a translation string (Col. 8, lines 30-41). The translation code stored in the translation table allows the invention in *Li* to determine a translation string for a given language code and translation code. In essence, the two tables form a lookup mechanism for determining what string to display for a graphical element given a language selected by a user and the translation code for the element.

The term string, as used in the present invention, is a series of characters manipulated as a group. In contrast, a database, or database table, is a collection of information organized in such a way that a computer program may quickly select desired pieces of data. Essentially, a database is an electronic filing system. Typical databases are organized by *fields*, *records*, and *files*. A field is a single piece of information; a record is one complete set of fields; and a file is a collection of records. For example, a telephone book is analogous to a file. It contains a list of records, each of which consists of three fields: name, address, and telephone number.

Further differences may be seen between strings and databases by examining the manner in which each is accessed. Below are figures illustrating the means in which a programmer utilizes a string and database table.

Figure 1 – String (put this stuff in a box)

```
string string_var;                  // declare the variable
string_var = "this is a string"; // assign a value to the variable
print_str(string_var);              // pass the variable to a function
```

Figure 2 – Database Table

```
database      database_var;                  // declare database variable
record_set    recordset_var;                 // declare recordset var to hold results
database_var.Open(<database name>);          // connect to the database
recordset_var = database_var.Execute(<SQL statement>);  // extract or set values in the database
```

String variables are considerably simpler to work with and manipulate compared to databases using modern programming languages such as C++ or Java. Generally, strings only exist as memory variables or as data stored in a file, such as a database table. Further, strings may be passed as a parameter to function, whereas a database table may not. A pointer to a database table may be passed to a function, but the database will still reside on disk. Moreover, databases function as a shared data repository and are access by many users, resulting in large size and slower access compared to string variables.

The invention in the present application is directed to a novel "text string data structure" which includes multiple fields wherein the content of each field is specified as set forth within the claims. A "text string" is a term of art in the computer technology area and as evidence of this fact, Applicant submits herewith page 451 of the Microsoft Press Computer Dictionary, Third Edition, which defines "string" as "a data structure composed of a sequence in characters usually representing human-readable text." Further evidence of the uniqueness of this verbiage is set forth within *Li, et al.*, at column 2, lines 23, *et seq.*, wherein *Li, et al.*, state "as an example, a PC-based attendant console application includes many embedded text strings that could not be intercepted by these types of multiple language systems." (emphasis added) It is precisely this problem which is addressed by the present Application in that translation system such as those disclosed within *Li, et al.*, are well known; however, such systems require a separate translation database to transform imbedded text strings found within an application or operating system. It is an object of the present Application to obviate that necessity by providing a novel, useful and unobvious "text string data structure" which includes multiple fields where each field includes a different content, as set forth within the present claims.

Specifically, with respect to Claim 1, the multi-field text string data object is expressly recited as including a first field "containing a first character string representing a word" and a second field "containing a second character string representing the word." In support of the rejection, the Examiner cites Table 2 of *Li, et al.*, which illustrates, in graphic form, the transform database utilized by the translation program of *Li, et al.* This transform database contains three fields, one field of which identifies a particular language. A second field of which contains a particular translation code which corresponds to an existing text string and the third field corresponds to a translation of that existing text string into the language identified by the language code. Applicant respectfully urges that a table such as that set forth within *Li, et al.*, at column 8, lines 42 *et seq.*, cannot be said to anticipate a "text string data structure" as a "text string data structure" is a specific term of art within the computer area and cannot be modified by the Examiner for purposes of applying a graphic representation of a transform database as disclosed within *Li, et al.* Further, a table does not show or suggest a "text string data structure" as each is a fundamentally different data type. As previously mentioned, the method of use and the abilities of strings and tables differ substantially.

Regarding the second group of claims, the Examiner cites *Renegar*, U.S. Patent No. 6,024,571 in combination with *Li, et al.*, for a rejection of Claims 4-11, 14-18 and 21-25 under 35 U.S.C. § 103(a), and Applicants contend such rejection is not well founded.

Renegar discloses a printed translation assistance learning aid which includes printed indicia including words and phrases in a first and second language and a phonetic pronunciation of those words. Nothing within *Renegar* shows or suggests in any way a "text string data structure" which is embedded within a computer usable medium as set forth within the present claims and no combination of *Renegar* with *Li, et al.*, can be said to show or suggest this novel data structure.

Further, as set forth expressly within Claims 10 and 16, the claims of Group 2, the third character string within the text string data structure is expressly recited as being prefixed by "at least one character with a low sort value." The Examiner has referred expressly to these claims and directs the Applicant's attention to the rejection of Claim 6; however, the Applicant direct attention to the fact that the rejection of Claim 6, and the rejection of Claims 10 and 16, as well as *Li, et al.*, and *Renegar* are completely silent with respect to any suggestion of the prefixing of a third character string by "at least one character with a low sort value" and the Board is urged to consider this rejection is not well founded.

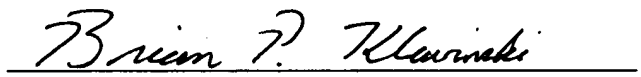
The Examiner has addressed this argument noting that *Li, et al.*, discloses the utilization of an ampersand, a character before the translation string "aide" believing that this constitutes "at least one character with a low sort value . . . ." In response to this position, Applicant urges the Board to consider that *Li, et al.*, itself is expressly contrary to the position taken by the Examiner. The ampersand symbol, at column 9, lines 4, *et seq.*, is described as being utilized to indicate which character within a given word or phrase is presented as underlined to the user for use as a so-called "hot key" or accelerator. Those having ordinary skill in the art will appreciate that this character will thus not be utilized in any type of sort operation and is merely present within the text string to indicate which of the characters will select that command when entered as a "hot key." The Examiner's position with respect to the "low sort value" of the ampersand signal is therefore not believed to have merit.

Consequently, Applicant urges that, Claims 10 and 16, contain features which are entirely absent from the references cited by the Examiner and that the remaining claims also define patentable subject matter over this combination of references and that the Examiner's rejection should be reversed.

Further, Applicant urges the Board to consider that data structures have been repeatedly found to be structure which must be considered in determining the patentability of an invention. *See In Re Lowry*, 32 U.S. P.Q.2d 1031 (Fed. Cir. 1994), wherein Applicant's invention concerned a data structure for storing, using and managing data in a computer memory. Similarly, *In Re Warmerdam*, 31 U.S. P.Q.2d 1754 (Fed. Cir. 1994), also addresses the subject of data structures. Applicant respectfully urges that the "text string data structure" of the present invention is neither shown nor suggested by *Li, et al.* or *Renegar*, whether considered alone or in combination as no combination of these references shows or suggests a text string data structure having multiple fields for representing a word such that the text string data structure may be manipulated within a data processing system and, by selecting one field or the other, the user may operate in one of two selected languages. Consequently, Applicant urges that Claims 1-25 define patentable subject matter over this combination of references and reversal of the Examiner's rejection is respectfully requested.

Please charge IBM Corporation Deposit Account No. **09-0447** in the amount of $320.00 for submission of a Brief in Support of an Appeal. No additional fees or expenses are believed to be required; however, if any additional fees are required, please charge IBM Corporation Deposit Account No. **09-0447**.

Respectfully submitted,



Brian P. Klawinski
Reg. No. *51,087*
Bracewell & Patterson, L.L.P.
Suite 350 Lakewood on the Park
7600B North Capital of Texas Highway
Austin, Texas 78731
(512) 542-2100

ATTORNEY FOR APPLICANT

APPENDIX

1. A text string data structure within a computer usable medium, comprising:

a multi-field data object encapsulating a plurality of discrete fields;

a first field within the multi-field data object containing a first character string representing a word; and

a second field within the multi-field data object containing a second character string representing the word.

2. The text string data structure of claim 1, wherein the second character string is different from the first character string.

3. The text string data structure of claim 1, wherein the first character string contains characters from a first character set employed by a first human language and the second character string contains characters from a second character set employed by a second human language.

4. The text string data structure of claim 1, wherein the first character string contains characters for a first human language and the second character string contains characters for a second human language which sound-map to characters within the first character string.

5. The text string data structure of claim 1, wherein the first character string contains an ideograph and the second character string contains a phonetic spelling of the ideograph. have symbols

6. The text string data structure of claim 1, further comprising:

a third field within the multi-field data object containing a third character string representing the word.

7. The text string data structure of claim 6, wherein the third character string is different from the second character string.

8. The text string data structure of claim 7, wherein the third character string is different from the first character string.

9. The text string data structure of claim 6, wherein:

the first character string contains characters for a first human language;

the second character string contains characters for a second human language which sound-map to characters within the first character string; and

the third character string is identical to the first character string.

10. The text string data structure of claim 6, wherein:

the first character string contains characters for a first human language; and

the third character string contains the first character string prefixed by at least one character with a low sort value.

11. The text string data structure of claim 6, wherein:

the first character string contains an ideograph;

the second character string contains Latin characters for a phonetic spelling of the ideograph; and

the third character string contains syllabary characters for a phonetic spelling of the ideograph.

12. A method of encapsulating information in a text string data structure, comprising:

creating a multi-field data object encapsulating a plurality of discrete fields;

storing a first character string representing a word in a first field within the multi-field data object; and

storing a second character string representing the word in a second field within the multi-field data object.

13. The method of claim 12, wherein the step of storing a second character string representing the word in a second field within the multi-field data object further comprises:

if the first character string contains characters from a first character set employed by a first human language, storing characters from a second character set employed by a second human language in the second field, wherein the second character string is different from the first character string.

14. The method of claim 12, further comprising:

storing a third character string representing the word in a third field within the multi-field data object.

15. The method of claim 14, further comprising:

storing characters from a first human language as the first character string;

storing characters from a second human language which sound-map to characters within the first character string as the second character string; and

storing characters identical to the first character string as the third character string.

16. The method of claim 14, further comprising:

storing the first character string prefixed by at least one character with a low sort value as the third character string.

17. The method of claim 14, further comprising:

storing an ideograph as the first character string;

storing a latin character phonetic spelling of the ideograph as the second character string; and

storing syllabary characters for a phonetic spelling of the ideograph as the third character string.

18. The method of claim 14, further comprising:

storing identical characters as the first, second, and third character strings.

19. A system for encapsulating information in a text string data structure, comprising:

means for creating a multi-field data object encapsulating a plurality of discrete fields;

means for storing a first character string representing a word in a first field within the multi-field data object; and

means for storing a second character string representing the word in a second field within the multi-field text string object.

20. The system of claim 19, wherein the means for storing a second character string representing the word in a second field within the multi-field data object further comprises:

means, if the first character string contains characters from a first character set employed by a first human language, for storing characters from a second character set employed by a second human language in the second field, wherein the second character string is different from the first character string.

21. The system of claim 19, further comprising:

means for storing a third character string representing the word in a third field within the multi-field data object.

22. The system of claim 21, further comprising:

means for storing characters from a first human language as the first character string;

means for storing characters from a second human language which sound-map to characters within the first character string as the second character string; and

means for storing characters identical to the first character string as the third character string.

23. The system of claim 21, further comprising:

means for storing the first character string prefixed by at least one character with a low sort value as the third character string.

24. The system of claim 21, further comprising:

means for storing an ideograph as the first character string;

means for storing a latin character phonetic spelling of the ideograph as the second character string; and

means for storing syllabary characters for a phonetic spelling of the ideograph as the third character string.

25. The system of claim 21, further comprising:

means for storing identical characters as the first, second, and third character strings.

streaming tape \strē´mēng tāp`\ *n.* *See* tape (definition 1).

stream-oriented file \strēm`ōr`ē-ent-əd fīl´\ *n.* A file used to store a fairly continuous series of bits, bytes, or other small, structurally uniform units.

street price \strēt´ prīs\ *n.* The actual retail or mail-order price of a consumer hardware or software product. In most cases, the street price is somewhat lower than the "suggested retail price."

stress test \stres´ test\ *n.* A test of a software or hardware system's functional limits, performed by subjecting the system to extreme conditions, such as peak volumes of data or extremes in temperature.

strikethrough \strīk´thro͞o\ *n.* One or more lines drawn through a selected range of text, usually to show deletion or the intent to delete, as in ~~strikethrough~~.

string \strēng\ *n.* A data structure composed of a sequence of characters usually representing human-readable text.

string variable \strēng` vâr´ē-ə-bl\ *n.* An arbitrary name assigned by the programmer to a string of alphanumeric characters and used to reference that entire string. *See also* string.

strobe \strōb\ *n.* A timing signal that initiates and coordinates the passage of data, typically through an input/output (I/O) device interface, such as a keyboard or printer.

stroke \strōk\ *n.* **1.** In data entry, a keystroke—a signal to the computer that a key has been pressed. **2.** In typography, a line representing part of a letter. **3.** In paint programs, a "swipe" of the brush made with the mouse or keyboard in creating a graphic. **4.** In display technology, a line created as a vector (a path between two coordinates) on a vector graphics display (as opposed to a line of pixels drawn dot by dot on a raster graphics display).

stroke font \strōk´ font\ *n.* A font printed by drawing a combination of lines rather than by filling a shape, as with an outline font. *Compare* outline font.

stroke weight \strōk´ wāt\ *n.* The width, or thickness, of the lines (strokes) that make up a character. *See also* font.

stroke writer \strōk´ rī`tər\ *n.* In video, a display unit that draws characters and graphic images as sets of strokes—lines or curves connecting points—rather than as sets of dots, as on a typical raster-scan monitor. *See also* vector graphics.

strong typing \strong` tī´pēng\ *n.* A characteristic of a programming language that does not allow the program to change the data type of a variable during program execution. *See also* data type, variable. *Compare* weak typing.

structure \struk´chur\ *n.* **1.** The design and composition of a program, including program flow, hierarchy, and modularity. **2.** A collection of data elements. *See also* data structure.

structured graphics \struk`churd graf´iks\ *n.* *See* object-oriented graphics.

structured programming \struk`churd prō´gram-ēng\ *n.* Programming that produces programs with clean flow, clear design, and a degree of modularity or hierarchical structure. *See also* modular programming, object-oriented programming. *Compare* spaghetti code.

structured query language \struk`churd kwēr´ē lang-wəj, kwâr´ē\ *n.* A database sublanguage used in querying, updating, and managing relational databases—the de facto standard for database products. *Acronym:* SQL (sē´kwəl, S`Q-L´).

structured walkthrough \struk´churd wäk`-thro͞o\ *n.* **1.** A meeting of programmers working on different aspects of a software development project, in which the programmers attempt to coordinate the various segments of the overall project. The goals, requirements, and components of the project are systematically reviewed in order to minimize the error rate of the software under development. **2.** A method for examining a computer system, including its design and implementation, in a systematic fashion.

STT \S`T-T´\ *n.* *See* Secure Transaction Technology.

stub \stub\ *n.* A routine that contains no executable code and that generally consists of comments describing what will eventually be there; it is used as a placeholder for a routine to be written later. *Also called* dummy routine. *See also* top-down programming.

StuffIt \stuf´it\ *n.* A file compression program originally written for the Apple Macintosh, used for storing a file on one or more disks. Originally shareware, StuffIt is now a commercial product for Macs and PCs that supports multiple compression techniques and allows file viewing. StuffIt files can